Title:	MCNP6.3 Executions in Parallel on Snow
Author(s):	Armstrong, Jerawan Chudoung
Intended for:	2023 MCNP User Symposium, 2023-09-18/2023-09-21 (Los Alamos, New Mexico, United States)
Issued:	2023-09-21 (rev.1)

MCNP6.3 Executions in Parallel on Snow

Jerawan Armstrong

2023 MCNP User Symposium
September 18-21, 2023

LA-UR-23-30411

Outline

- Parallel Computing in MCNP6
- Test Problem I: Athena-I Model
- Test Problem II: CANDU Model

Parallel Computing in MCNP6

Three Options for Running MCNP6 in Parallel

- mpirun –np X mcnp6.mpi i=filename

- mcnp6 i=filename tasks N

- mpirun –np X mcnp6.mpi i=filename tasks N

“**mpirun –np**” should be replaced with “**srun –n**” on High Performance Computing (HPC) machines using Slurm.

Which option should be used to run MCNP6 in parallel?

Parallel Computing

Running Parallel Simulations

Parallel Computers
(hardware)

+

Parallel Codes
(software)

```
!$OMP PARALLEL
call trnspt
!$OMP END PARALLEL
```

TASKS option

```
call msg_send(0, 1002, fTrack(i)%noEls)
call msg_send(0, 1002, fTrack(i)%link)
call msg_send(0, 1002, fTrack(i)%link2)
call msg_send(0, 1002, fTrack(i)%cutDir)
call msg_send(0, 1002, fTrack(i)%sboxs)
call msg_send(0, 1002, fTrack(i)%cents)
call msg_send(0, 1002, fTrack(i)%elNum)
```

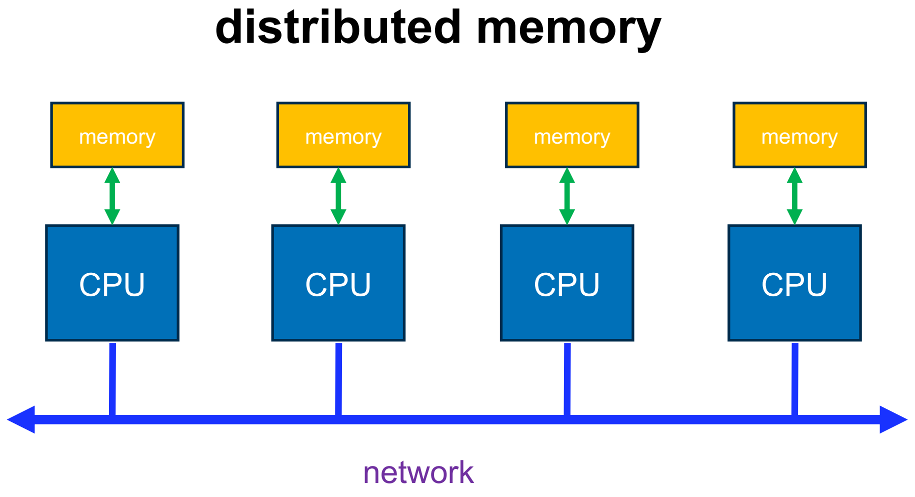
MPIRUN option

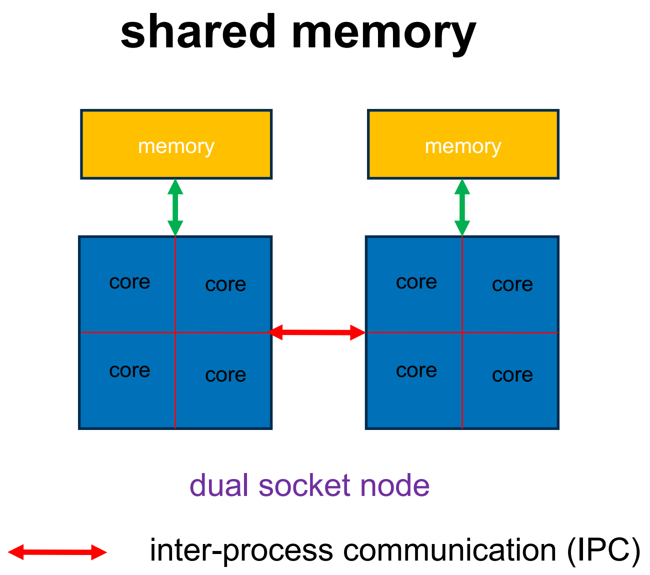
```
call msg_recv(mn, 1002, fTrack(i)%link)
call msg_recv(mn, 1002, fTrack(i)%link2)
call msg_recv(mn, 1002, fTrack(i)%cutDir)
call msg_recv(mn, 1002, fTrack(i)%sboxs)
call msg_recv(mn, 1002, fTrack(i)%cents)
call msg_recv(mn, 1002, fTrack(i)%elNum)
```

Parallel codes implemented in MCNP6

Parallel code implementation is strongly influenced by a parallel computing system to be used.

General Parallel Architecture

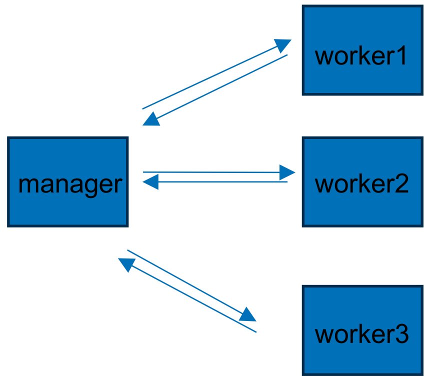




- CPUs may consist of one or more **cores,** a distinct execution unit with its own instruction stream. CPU cores may be organized into one or more **sockets** where each socket has its own distinct memory. When a CPU consists of multiple sockets, memory sharing across sockets is usually supported by hardware infrastructure [1].
- A **symmetric multi processor (SMP)** is a shared memory architecture where multiprocessors share a single address space and have equal access to all resources [1].
- A **node** is a standalone computer**,** consisting of CPU cores, memory, network interfaces, etc. Nodes are networked together to form a supercomputer [1].

MPI Parallel Computing in MCNP6

- Message Passing Interface (MPI) programming model was originally designed for distributed memory architectures, but presently MPI programs may run on distributed memory, shared memory, or hybrid (distributed-shared) memory systems [2].
  - Distributed memory: network-based memory access for physical memory that is not common.
  - Shared memory: all processors have direct access to common physical memory.
  - Hybrid memory: shared memory nodes are linked to form a cluster, and network communications are required to move data between nodes. A typical HPC machine is a cluster of SMP nodes.
- A command line for MPI parallel computing in MCNP6: **mpirun –np X mcnp6.mpi i=test** or **mpiexe –n X mcnp6.mpi i=test**
  - MCNP checks X. For MPI parallel computing, X >=3. If X=1, 2, no MPI parallel computing. 1 manager, and (X-1) worker tasks with 1 thread each

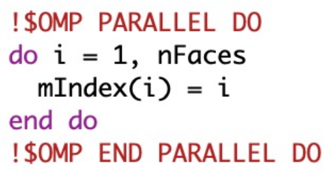


MPI-3 was used in MCNP6.3

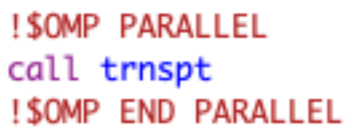
OpenMP Parallel Programming in MCNP6.3

- OpenMP is an API (Application Program Interface) for multi-threaded, shared memory parallel programming.
  - "A **thread** of execution is the smallest unit of processing that can be scheduled by an operating system [3].

- OpenMP parallel programing in MCNP is specified by using compiler directives.



```
!$OMP PARALLEL DO
do i = 1, nFaces
  mIndex(i) = i
end do
!$OMP END PARALLEL DO
```



```
!$OMP PARALLEL
call trnspt
!$OMP END PARALLEL
```

- A command line for an OpenMP parallel computing in MCNP6: **mcnp6 i=test tasks N**

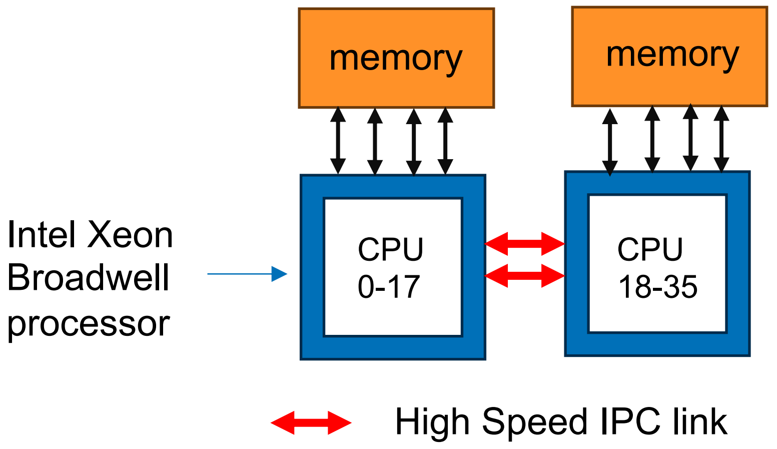
An OpenMP only option should not be used to run MCNP on multiple nodes of HPC clusters.

Parallel Programming Improvements in MCNP6.3 Unstructured Mesh (UM) Feature

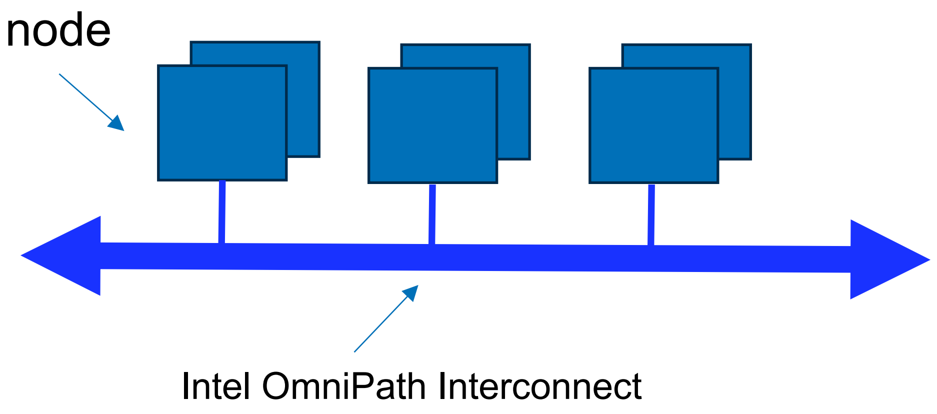
- Used MPI-3.
- Added OpenMP parallel directives in codes used for input processing.
- Rewrote codes used for parallel input processing.
  - An MPI execution for UM input processing only option is allowed.
- Fixed MPI and OpenMP bugs.

Snow

- Snow is a LANL’s Commodity Technology System Phase I (CTS-1) cluster.
  - 368 compute nodes, Intel Xeon Broadwell processor & Intel OmniPath interconnect.
  - Each node has 36 CPU cores with 128GB memory (~3.5 GB/core) where cores are grouped into 2 sockets.
  - Each CPU core has two hyperthreads which are disabled.
- Snow uses the **Slurm** Workload Manager for resource allocation and scheduling, work monitoring, queue management.
- How to run calculations in parallel on HPC machines is problem-dependent [4].



node

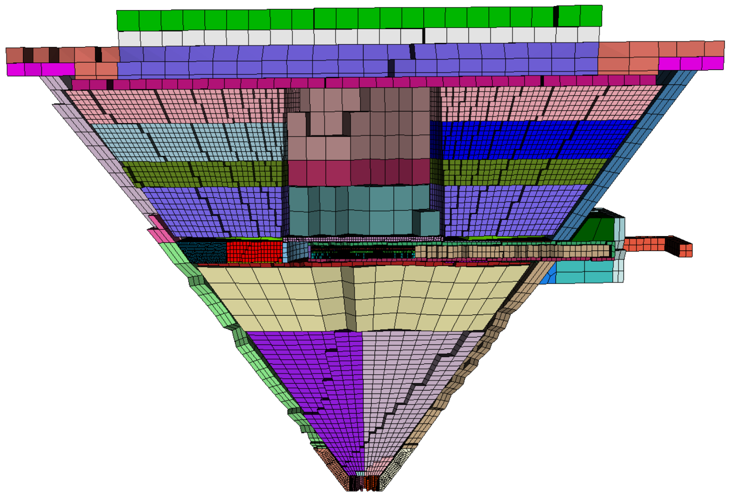


A cluster of nodes

Test Problem I: Athena-I Model

Athena-I Model

- An Athena-I experiment was designed by the Air Force Institute of Technology (AFIT) and conducted at the National Ignition Facility (NIF).
- Hybrid geometry: constructive solid geometry (CSG) & unstructured mesh (UM) geometry; 1,107,965 linear hexahedral elements & 43 parts.
- Fixed source problem, MODE n p
- NPS 1E8
- PRDMP 1E9 1E9 1 2 1E9


An Athena-I hexahedral model was created by Bradley Gladden [5].

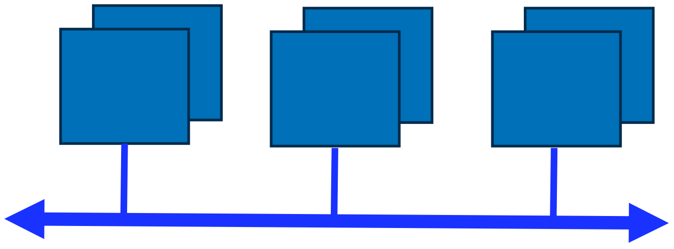
Athena-I: MPI on Snow

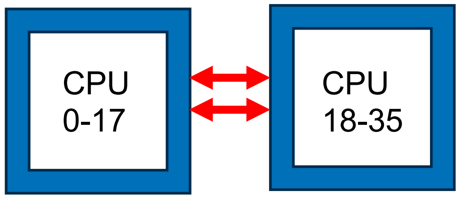
6 runs: srun –n X mcnp6.mpi i=athena1_hex.txt

X-1 MPI worker tasks with 1 thread each

node	cores X	histories/ hr	computer time [minutes]	wall clock time [h:m:s]
1	36	59.72E6	3617.40	01:41:17
2	72	117.78E6	3669.43	00:41:53
4	144	238.23E6	3630.15	00:26.01
6	216	352.68E6	3679.99	00:17.55
8	288	471.41E6	3672.76	00:13.34
10	360	590.35E6	3668.59	00:10.58

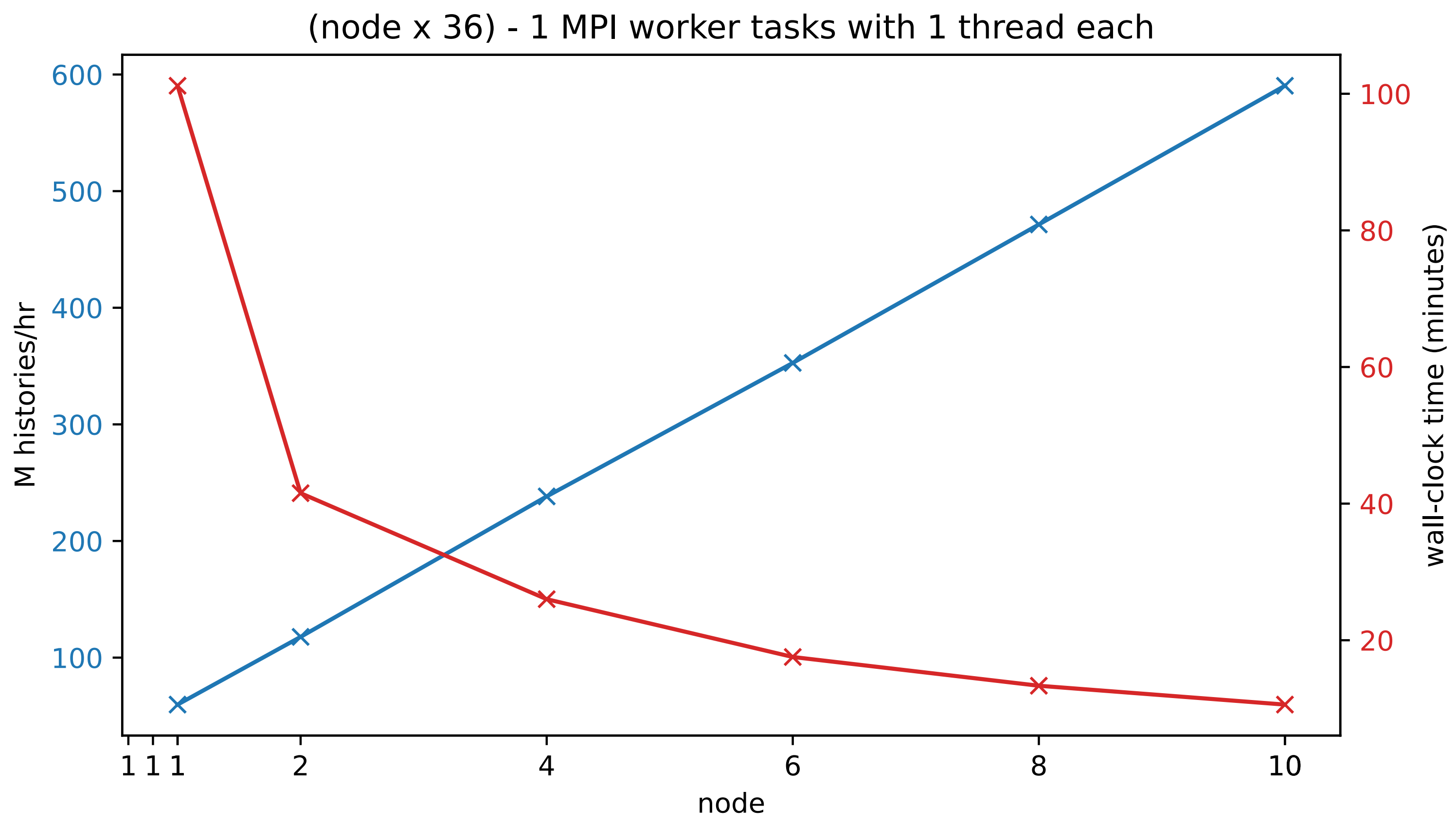
- After a job is submitted to Slurm, it will be in the queue before being executed on the compute nodes.
- Goal: choose a number of nodes that the queue time and wall clock time are minimized.
- Generally, the more nodes a user requests, the more times a job will be in the queue.





36 cores per node

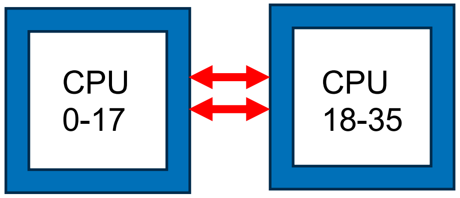
Athena-I: MPI on Snow



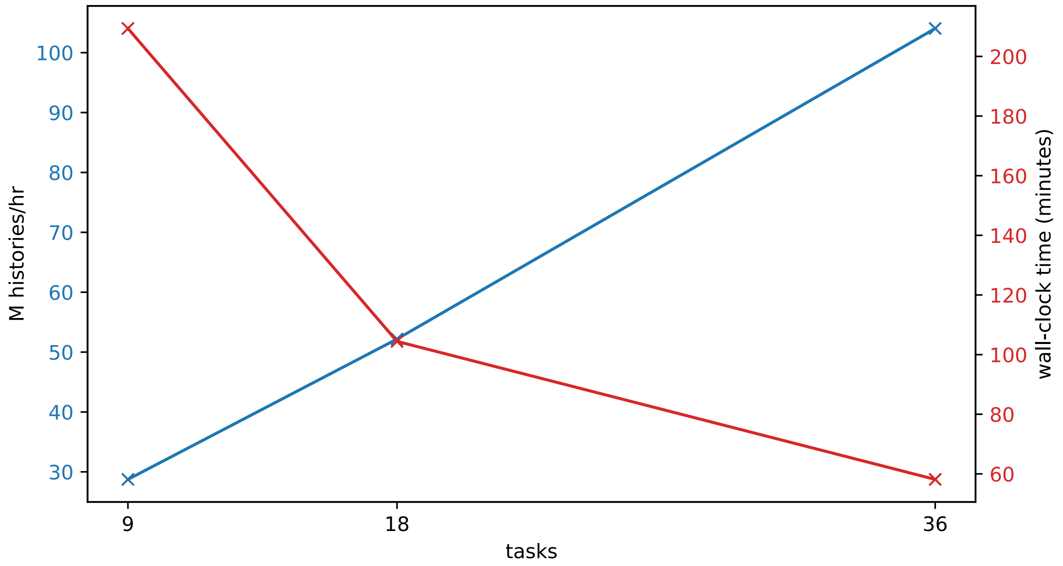
Athena-I: OpenMP on Snow Using a Single Node

3 runs on 1 node: mcnp6 i=athena1_hex.txt tasks N

N	histories/hr	computer time [minutes]	wall clock time [h:m:s]
9	28.74E6	1879.78	03:29:37
18	52.18E6	2070.23	01:44:44
36	104.04E6	2077.23	00:58:15



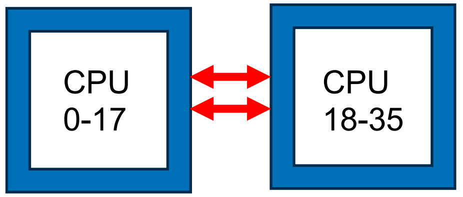
36 cores per node



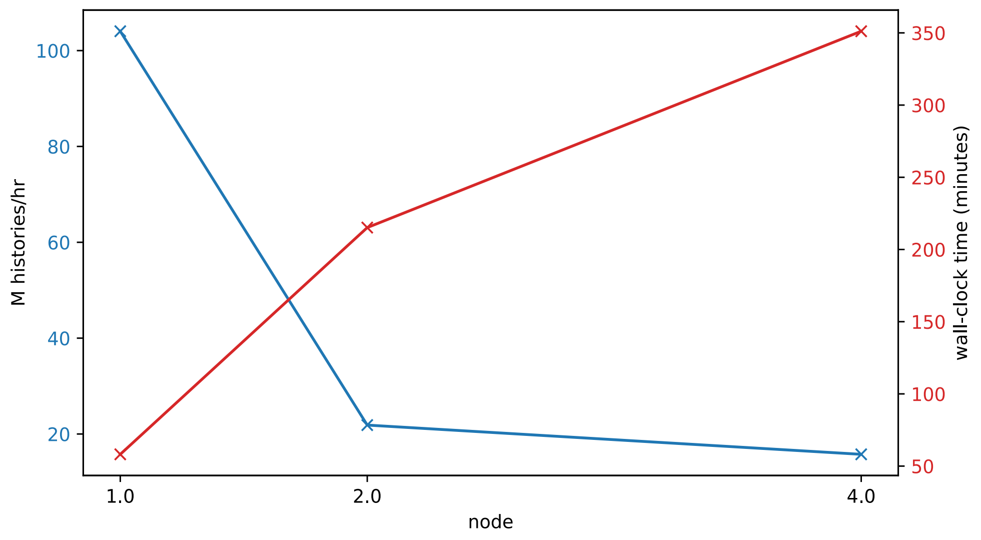
Athena-I: OpenMP on Snow Using Multiple Nodes

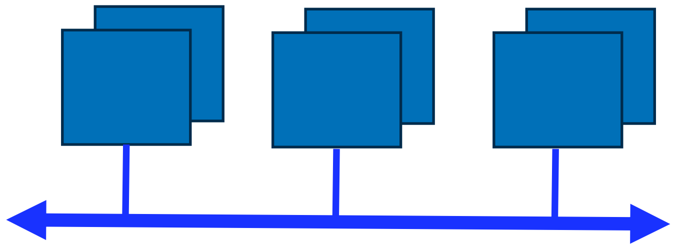
3 runs: mcnp6 i=athena1_hex.txt tasks N

nodes	cores N	histories/hr	computer time [minutes]	wall clock time [h:m:s]
1	36	104.04E6	2077.23	00:58:15
2	72	21.86E6	9882.51	03:35:17
4	144	15.76E6	13703.17	05:51:14



36 cores per node





A pure multithreading (OpenMP) parallelization option should only be used on a single compute node.

Athena-I: Performance on 1 Compute Node

```
srun –n 36 mcnp6.mpi i=athena1_hex.txt
```

node	cores	histories/hr	computer time [minutes]	wall clock time [h:m:s]
1	36	59.72E6	3617.40	01:41:17

```
mcnp6 i=athena1_hex.txt tasks 36
```

nodes	cores	histories/hr	computer time [minutes]	wall clock time [h:m:s]
1	36	104.04E6	2077.23	00:58:15

- MPI+OpenMP may be a reasonable option for running this problem using multiple nodes since the performance of OpenMP computing is better than the performance of MPI computing on a single compute node.

In general, a user should use a pure MPI and OpenMP option on a single node to check the performance. If the performance of OpenMP computing is better than the performance of MPI computing, then a hybrid (MPI+OpenMP) option could be used to run the problem using multiple compute nodes.

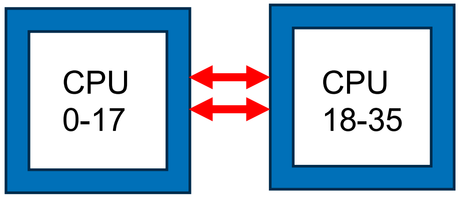
Athena-I: MPI + OpenMP on Snow

4 runs: srun –n X mcnp6.mpi i=athena1_hex.txt tasks 36

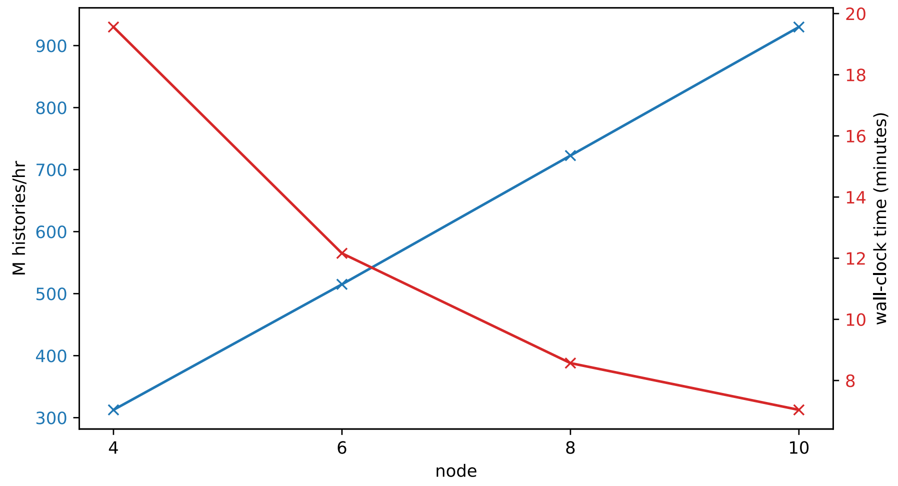
X-1 MPI worker tasks with 36 threads each

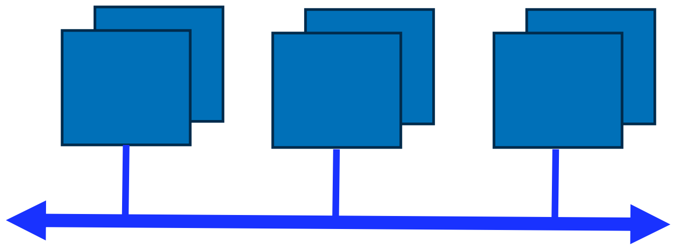
nodes X	cores	histories/hr	computer time [minutes]	wall clock time [h:m:s]
4	144	312.57E6	2088.76	00:19.56
6	216	515.03E6	2094.29	00:12:16
8	288	722.64E6	2087.11	00:08:57
10	360	929.94E6	2090.67	00:07:04

For MPI parallel computing in MCNP6, X must be >= 3.

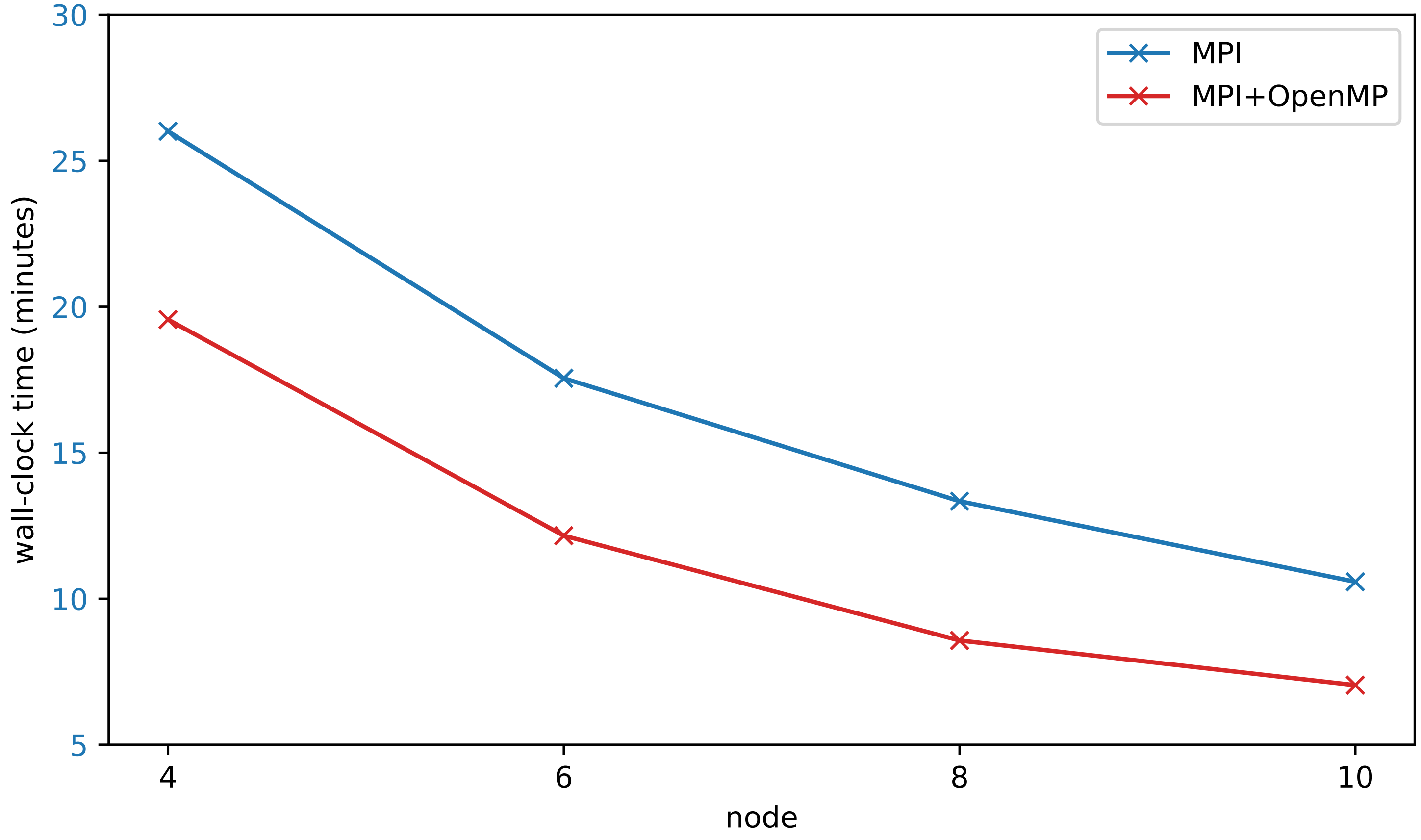


36 cores per node





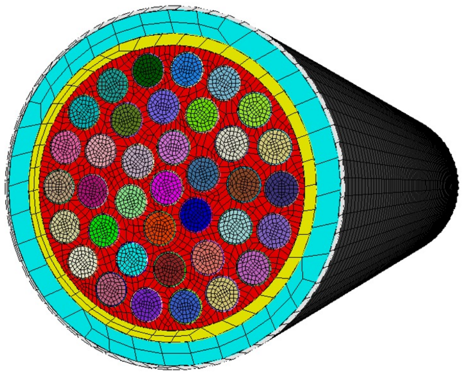
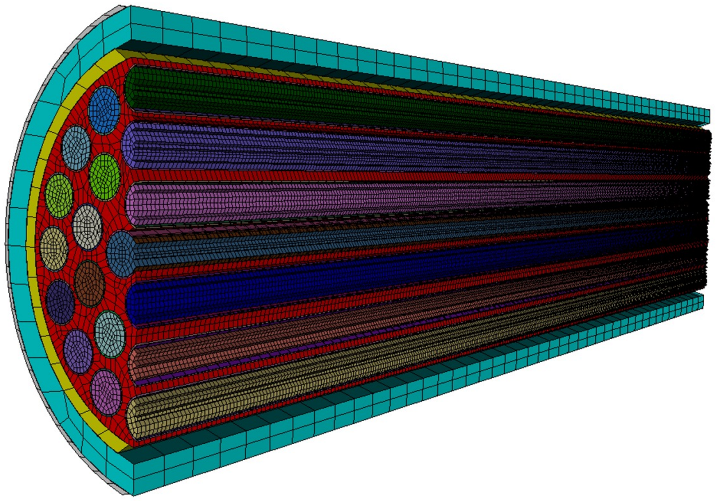
Athena-I: Wall Clock Time Comparison



Test Problem II: CANDU Model

CANDU Model

- Canadian Deuterium natural Uranium reactor fuel bundle 37-element.
- Hybrid geometry: constructive solid geometry (CSG) & unstructured mesh (UM) geometry; 2,716,982 linear tetrahedral and hexahedral elements & 5 parts.
  - 350,878 Tet elements & 2,366,104 Hex elements.
  - Parts were grouped by materials; 1 Tet part and 4 Hex parts.
- The UM bundle is inside a CSG cell which is reflected to representing a full core.
- KCODE problem; MODE n
  - KCODE 100000 1.0 30 150
- PRDMP 1E10 1E10 1 2
  - Use default 5th entry (dmmp=0), writing TFC entries 10 times total.

A CANDU model was created by Esteban Gonzalez [6].

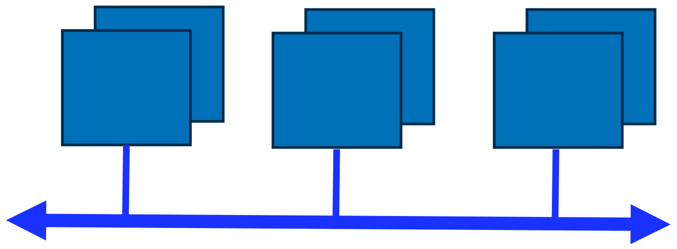
Los Alamos NATIONAL LABORATORY

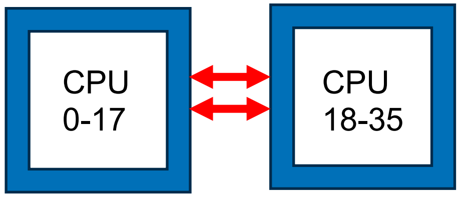
CANDU: MPI on Snow

6 runs: srun –n X mcnp6.mpi i=candu_hex.txt

X-1 MPI worker tasks with 1 thread each

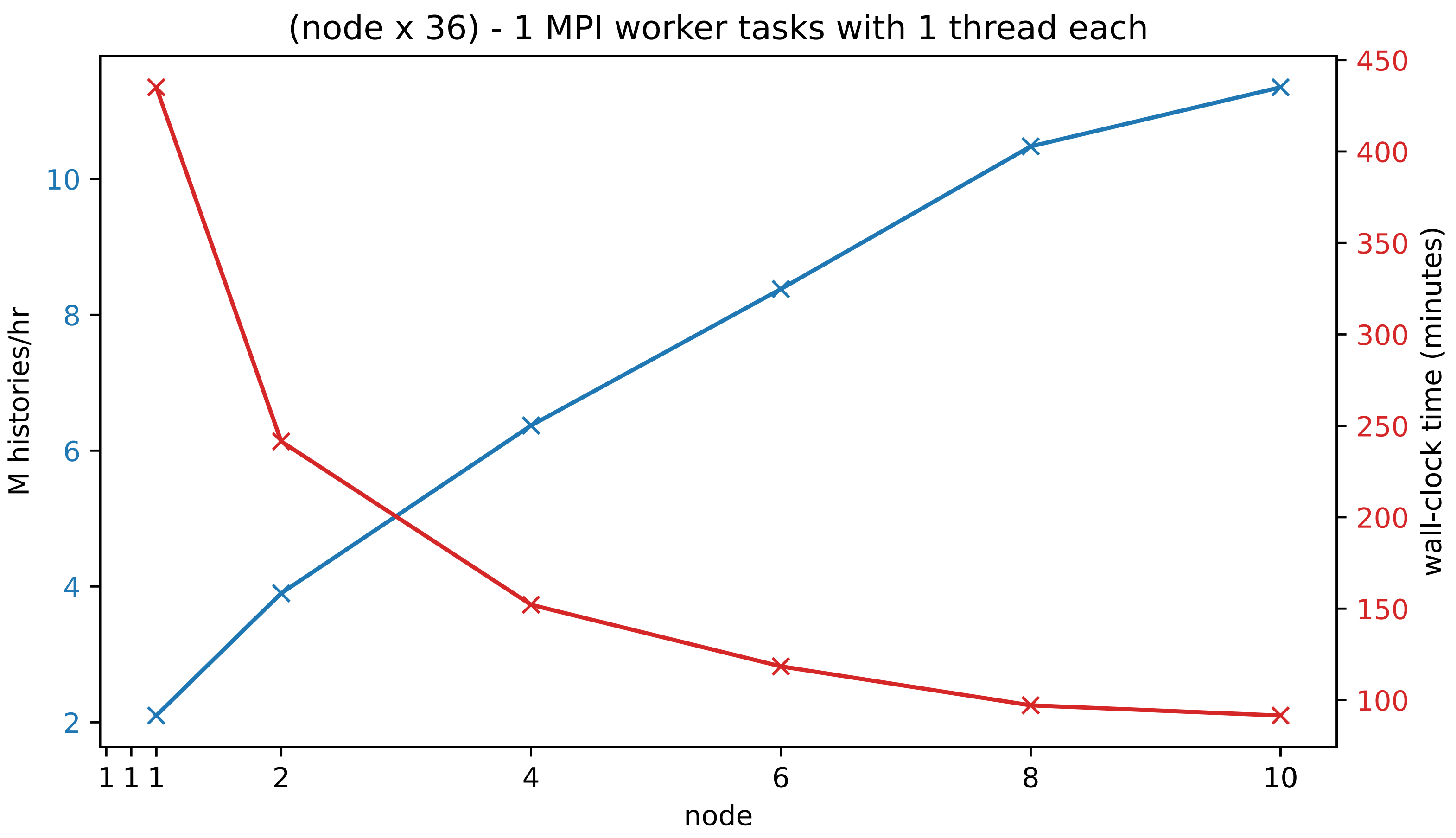
node	cores X	histories/hr	computer time [minutes]	wall clock time [h:m:s]
1	36	2.10E6	15319.81	07:15:13
2	72	3.90E6	16379.02	04:01:49
4	144	6.37E6	19659.16	02:32.15
6	216	8.38E6	22763.50	01:58.45
8	288	10.48E6	24094.55	01:37:14
10	360	11.35E6	27792.20	01:31:56





36 cores per node

CANDU: MPI on Snow



CANDU: OpenMP and MPI+OpenMP on Snow

- The performance of OpenMP computing was worse than the performance of MPI computing on a single compute node.

- When using an OpenMP option on a single compute node (mcnp6 i=candu_hex.txt tasks 36), the calculation was not finished in 12 hours.
  - Only 9 cycles were finished in 12 hours.

- **The hybrid (MPI + OpenMP) option should not be used to run this problem using multiple nodes.**
  - Using srun –n 8 mcnp6.mpi i=candu_hex.txt tasks 36, only 64 cycles were finished in 12 hours.
  - Using srun –n 10 mcnp6.mpi i=candu_hex.txt tasks 36, only 84 cycles were finished in 12 hours.

Conclusions

- Three options to run MCNP6 in parallel: MPI, OpenMP, MPI+OpenMP
  - Which option to use depends on a computer to be used and a problem to be run.
- MPI option:
  - Works on distributed memory, shared memory, or hybrid (distributed + shared) memory architectures.
  - Scalable. Using more compute nodes may reduce walk clock times.
- OpenMP option:
  - Works on shared memory architectures.
  - Should not be used to run on multiple compute nodes of distributed memory machines.
  - Not scalable. It is very rare to have a large shared memory computer.
- MPI+OpenMP option:
  - Works on hybrid (distributed + shared) memory architectures.
  - Should not be used for a problem where the performance of OpenMP option is worse than the performance of MPI option.
- Should perform two short runs to compare the performances of pure MPI and OpenMP calculations before using a hybrid (MPI+OpenMP) option on multiple compute nodes.
- Should use PRDMP card for parallel runs.

Questions?

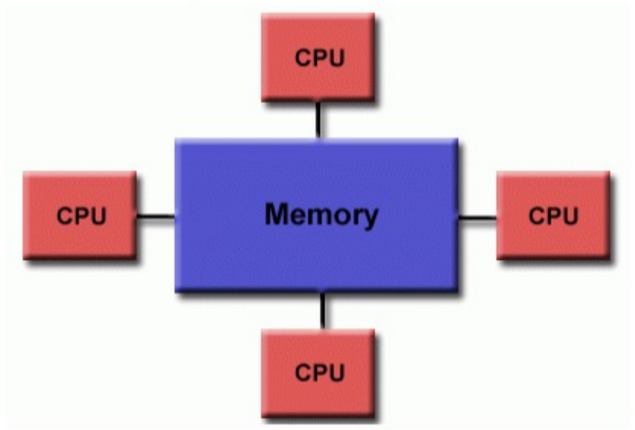
References

1. https://hpc.llnl.gov/documentation/tutorials/introduction-parallel-computing-tutorial
2. https://hpc-tutorials.llnl.gov/mpi
3. https://hpc-tutorials.llnl.gov/openmp
4. https://researchcomputing.princeton.edu/support/knowledge-base/scaling-analysis
5. Bradley Gladden et al. “Athena-I CUBIT Journal Files”, Tech. rep. LA-UR-23-28395. Los Alamos NM, USA: Los Alamos National Laboratory, Jul. 2023. https://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-23-28395
6. Esteban Gonzalez et al. “MCNP6.3 Unstructured Mesh Verification: GodivaR and CANDU Models”, Tech. rep. LA-UR-22-33091. Los Alamos NM, USA: Los Alamos National Laboratory, Dec. 2022. https://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-22-33091

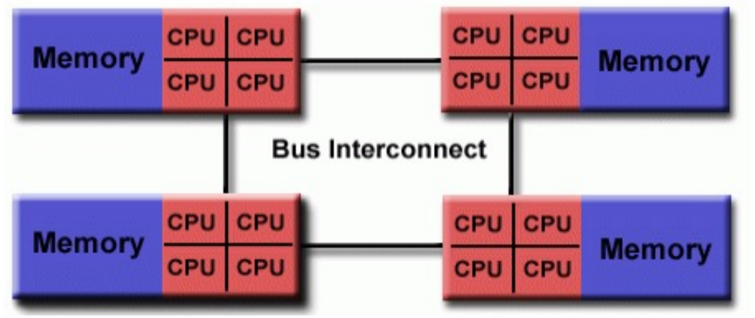
Legal Notice

Parallel Computer Memory Architectures

Shared Memory

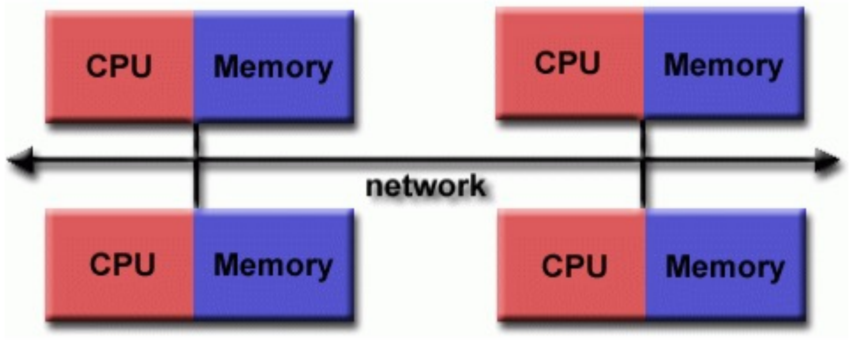


Uniform Memory Access (UMA)

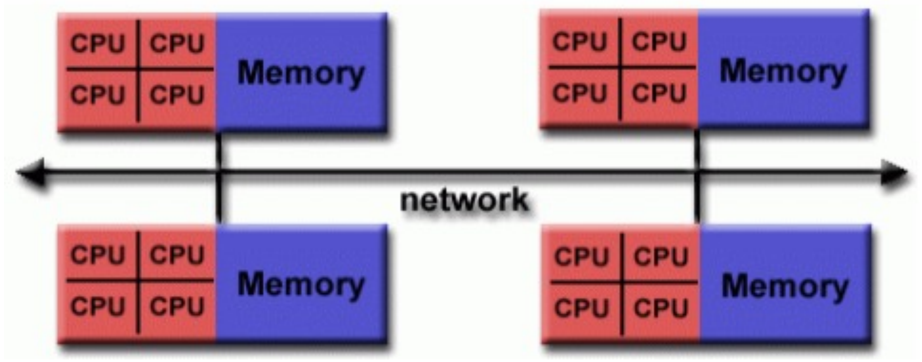


Non-Uniform Memory Access (NUMA)

Distributed Memory



Hybrid Distributed-Shared Memory



source: https://hpc.llnl.gov